1994
NASA/ASEE SUMMER FACULTY FELLOWSHIP PROGRAM
MARSHALL SPACE FLIGHT CENTER

THE UNIVERSITY OF ALABAMA

ERROR CODING SIMULATIONS IN C

Prepared by:	Viveca K. Noble
Academic Rank :	Instructor
Institution and Department	Tuskegee University Department of Electrical Engineering
MSFC Colleagues:	Bernd K. Seiler Helen L. Thomas
NASA/MSFC:	
Office:	Astrionics Laboratory
Division:	Computers and Data Management
Branch:	Flight Data Systems

Introduction

When data is transmitted through a noisy channel, errors are produced within the data rendering it indecipherable. Through the use of error control coding techniques, the bit error rate can be reduced to any desired level without sacrificing the transmission data rate.[2]

The Astrionics Laboratory at Marshall Space Flight Center has decided to use a modular, end-to-end telemetry data simulator to simulate the transmission of data from flight to ground and various methods of error control. The simulator includes modules for random data generation, data compression, Consultative Committee for Space Data Systems (CCSDS) transfer frame formation, , error correction/detection, error generation and error statistics . The simulator utilizes a concatenated coding scheme which includes CCSDS standard (255,223) Reed-Solomon (RS) code over GF(2^8) with interleave depth of 5 as the outermost code, (7, 1/2) convolutional code as an inner code and CCSDS recommended (n, n-16) cyclic redundancy check (CRC) code as the innermost code, where n is the number of information bits plus 16 parity bits. The received signal-to-noise for a desired bit error rate is greatly reduced through the use of forward error correction techniques. Even greater coding gain is provided through the use of a concatenated coding scheme.[4] Interleaving/deinterleaving is necessary to randomize burst errors which may appear at the input of the RS decoder.[5] The burst correction capability length is increased in proportion to the interleave depth.[2] The modular nature of the simulator allows for inclusion or exclusion of modules as needed. This is a cost-effective means of determining optimal error control schemes for a given error distribution.

System Description, Initial Development and Results

A block diagram illustrating the operation of the simulator is shown in Figure 1.

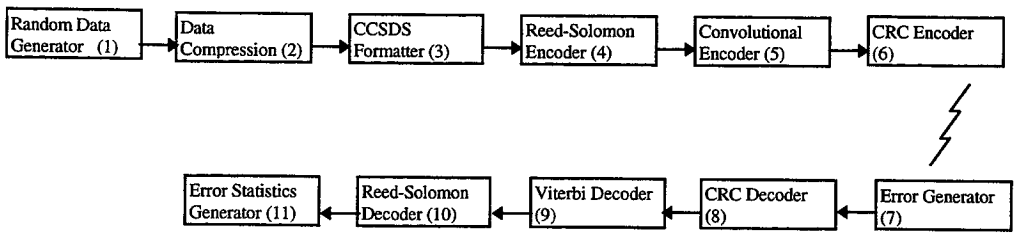

Figure 1

In the initial development phase of the simulator, modules (1), (3), (6), (7), (8) and (11) were developed in FORTRAN and the code simulating the CRC encoder and decoder shown in Figures 3 and 4, respectively, were verified for up to 3 random errors. The CCSDS formatter inserts a 32-bit sync marker ($1ACFFC1D_{hex}$) and stores the 48 bits immediately following the sync marker as header infomation as shown in Figure 2. The following recommendations and/or tasks resulted from this work: [4]

- Convert simulation programs from FORTRAN to C
- Determine appropriate error distributions
- Develop Reed-Solomon and convolutional code simulator programs
- Add data compression modules
- Develop more refined and flexible error generator program

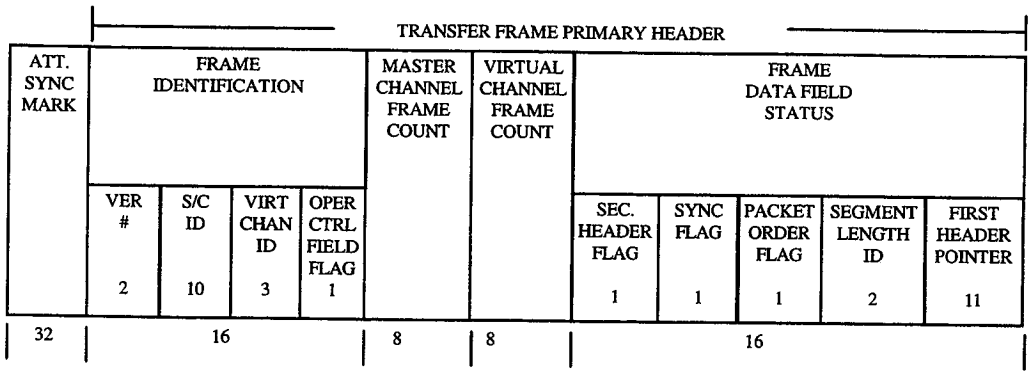

Figure 2

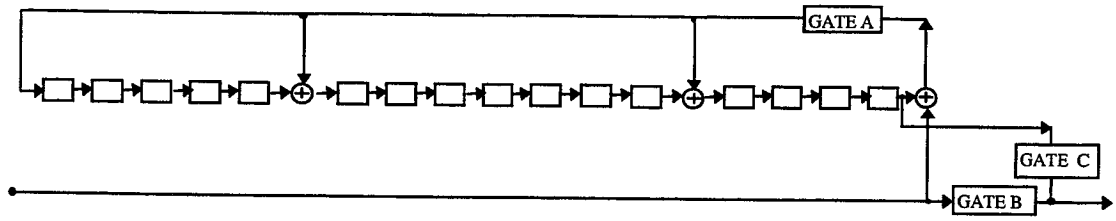

Figure 3

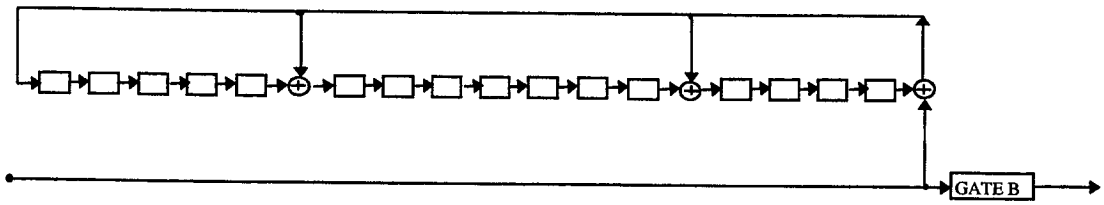

Figure 4

Intermediate Development and Results

Code which was written during the initial development phase has been converted to C due to the flexibility of the language. It was determined that random errors are represented by an additive white Gaussian distribution and bursts are represented by a Markov Chain model.[1]

The error generator that was developed initially was random in nature; in order to be "more refined and flexible", the error generator should possess the ability to generate random, burst and a combination of random and burst errors. Code written in C that simulates a RS encoder/decoder [Rockliff, Simon- University of Adelaide] was obtained from an Internet users' group. The testing of the code in various bit error rate environments is a continuous process due to the very purpose of the simulator which is to determine optimal error control schemes for a given error distribution.

Encoding for the RS code mentioned above is in systematic form and the Berlekamp iterative algorithm is used for decoding. The code may be modified to suit particular needs, that is, one may specify m (any positive integer), n (the length of the codeword), k (the length of the information string) and t (the number of errors that can be corrected). Also, the irreducible polynomial must be specified to generate the Galois field, $GF(2^m)$. These polynomials may be found in [2]. In its present form, the code does not handle erasures but may be modified to do so by using the Berlekamp-Massey algorithm. In addition, it does not attempt to decode beyond the BCH bound.[Rockliff, Simon - University of Adelaide]

Research regarding various error control coding and increases in coding gain revealed that that increases in coding gain resulted in increases in decoding complexity. The degree of decoding complexity far exceeded the degree to which the coding gain increased. Thus, hardware implementation of such a decoder was not justified.[3]

Exploration of Software Tools

In addition to developing its own telemetry data simulator, the Astrionics Laboratory is investigating the possibility of using Comdisco SPW™ in its efforts to determine optimal error control schemes. SPW™ (Signal Processing Worksystems™ - a trademark of Comdisco) has built-in libraries for data generation, noise generation, RS encoding and convolutional encoding. One may create various coding schemes by arranging and rearranging these software modules. During construction of the error control coding scheme, only block diagrams representing its components may be seen. However, one may "step down" by levels and view the circuit diagram of the encoders as well as the C code which was written to simulate each component. Since there are library functions for the components, encoders may also be "constructed" from circuit diagrams by simply drawing the circuit in the SPW™ environment. This software tool is phenomenal.

Conclusion and Future Tasks

The C code for the Reed-Solomon has been verified. Research regarding algorithms for burst error generation, data compression and convolutional encoding will continue. Once obtained, code for these modules will be written and verified. Once all modules are complete, they will be compared to their hardware equivalents, if application, to verify the correct operation of the software. Investigation into the possibility of using SPW™ to verify the software will continue.

References

[1] Berman, Ted and Dr. Jeffrey Freeman, *Non-interleaved Reed-Solomon Coding Over A Bursty Channel*, Communications - Fusing Command, Control and Intelligence Conference, Vol. 2, 1992, p. 580

[2] Lin, Shu, Daniel J. Costello, Jr., *Error Control Coding: Fundamentals and Applications*, Prentice Hall, Inc., Englewood Cliffs, N.J., 1983, pp. 1, 28, 272

[3] Lin, Shu, Daniel J. Costello, Jr., Warner H. Miller, James C. Morakis, William B. Poland, Jr., *Bandwidth Efficient Coding for Satellite Communications*, NAG 5-931 and NAG 5-557

[4] Noble, Viveca K., *Error Coding Simulations*, Final Report, Contract NASA, CR-193862, August 1993, p.1

[5] Siveski, Z., Bozovic, R., Schilling, D. L., *Concatenated Trellis/High-Rate Reed-Solomon and Projection Codes*, IEEE International Conference on Communications, Vol.1, 1989, p. 563